Knowledge Engineering and Data Science (KEDS)
Vol 1, No 1, January 2018, pp. 33–38

pISSN 2597-4602
eISSN 2597-4637

SQL Logic Error Detection using Start End Mid Algorithm

Jevri Tri Ardiansah [a, 1, *], Aji Prasetya Wibawa [a, 2],
Triyanna Widiyaningtyas [a, 3], Okazaki Yasuhisa [b, 4]

[a] *Department of Electrical Engineering, State University of Malang, Jl. Semarang No.5, Malang 65145, Indonesia*
[b] *School of Science and Engineering, Saga University, Honjo-machi, Saga 840-8502, Japan*
[1] *jev.ardian@gmail.com*;* [2] *aji.prasetya.ft@um.ac.id;* [3] *triyannaw@gmail.com;* [4] *okaz@cc.saga-u.ac.jp*
** corresponding author*

ARTICLE INFO

Article history:
Received 27 August 2017
Revised 13 September 2017
Accepted 3 November 2017
Published online 8 January 2018

Keywords:
SQL
Logic Error
String Matching
Start End Mid Algorithm

ABSTRACT

Database is an important part of a system and it stores data to be manipulated. SQL (Structured Query Language) is used for manipulating those data to extract information and make decision. There are two types of error which make SQL is challenging to learn, namely syntax error and logic error. Compiler can detect syntax error, but it does not show error warning while logical error occurred. It makes logic error more difficult to understand than syntax error. A web based SQL compiler with errors detection ability by using Start End Mid algorithm is then developed, To help database's user to learn SQL in practical implementation.

I. Introduction

SQL is an important language since it used to access database to extract information and make decision. There are two types of error that can be occurred in SQL syntax, they are syntax error and logic error [1]. Syntax error is caused by mistakes in syntax writing, so it is automatically detected by the compiler. In this case, compiler gives warning about occurred error to let users fix their mistakes. By those mechanisms, users can easily learn from given error messages. On the other hand, logic error occurs if the syntax is correct but it does not produce the intended result [2]. It makes logic error more difficult to learn since there is no warning from compiler about occurred mistakes [3].

Based on observation, SQL in theory is easy to understand but it is difficult to use SQL syntax in practical implementation [4]. That conclusion is supported by another research about SQL errors. By giving some questions to students and new programmer; let them answer with SQL syntax, the result shows that 32% participants made logical error, 8% made syntax error and 14% made both syntax error and logic error [5]. Based on that research, the possibility of logic error occurrence is 46%, it is larger than syntax error.

One of the ways to detect the occurred logical error is by knowing the task to detect the case whether it has solved or not go [6]. In this system, if compiler did not detect syntax error, result table will be produced. The result table can be used to check the task status. If it is different with the expected one, it means that the task's status is not solved yet. By that status, it can be concluded that users made a logical error. This potential way can be used to show the logical error information to users and let them correct the mistakes.

II. Methods

This system provides many cases based on selection keywords in SQL. There are 36 cases in total that should be answered using SQL syntax. User can choose any of the keywords based on what they want to learn. Each keyword contains material and case so users are able to learn and

https://doi.org/10.17977/um018v1i12017p33-38

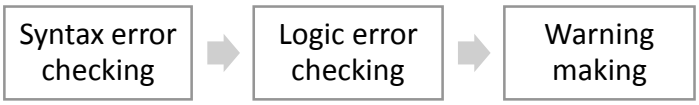

Fig. 1. The method to produce error warning

answer each case using SQL syntax. By submitting SQL query to the compiler, system shows the status of submitted query and the type of error; no error, syntax error or logic error.

The input of this system is SQL query and the output is a warning. The warning is about no error and syntax error or logic error which occured. From that warning, users are expected to apply correction and learn by their mistakes. System has three main steps to produce syntax or logical error information to users [7]. Those steps are shown as Fig.1.

The first step is syntax error checking. System uses online SQL compiler to detect syntax error. That compiler compiles query to web server by a real database management system. This system uses MySQL as database management system. After compiling process, the occured syntax error will be shown to warn user about their mistakes. After there is no syntax error, system will produce result table based on submitted query.

To check whether the query contains logic error or not, system compares the similarity between the result table and key table in the second step. In this case, key table is the expected table that should be produced by given task. If there is difference between them, then user did not answer the case correctly. It means that there is logic error in their answer query [6].

There are many string matching algorithms based on Brute Force algorithm. Those algorithms are Knuth-Morris-Pratt (KMP), Boyer-Moore (BM) and Karp and Rabin [8]–[10]. Start End Mid algorithm is also a developed algorithm based on Brute Force Algorithm [11]. Start End Mid algorithm is used to check the similarities between result table and key table. Result table is the table which is produced by user's SQL and key table is the correct table which is stored in data base. Start End Mid algorithm has many advantages in system. This algorithm is simple and easy because it has no preprocessing phase like other string matching algorithms [12], [13]. Those characteristics are very usefull in implementation considering this system uses small of data. This algorithm checks the first, end and mid character before doing sequential checking like Brute Force. Characters between result table and expected table will be put in array and it will be checked by this algorithm. The steps in similarity checking based on Start End Mid algorithm are:

a. Compare the first character between result table and expected table. If they are similar, then go to the next step.
b. Compare the last character between result table and expected table. If they are similar, then go to the next step.
c. Compare the middle character between result table and expected table. If they are similar, then go to the next step.
d. Compare remaining characters from start to end sequentially.

Using the steps outlined above, it is possible to measure whether there is a difference; even in a single step. If there is a mistake, then there is a difference between result table and expected table [14], [15]. It means that user did not answer the related case correctly. In this case, they made a logic error in their submitted SQL syntax.

After logic error is detected, the system gives warning to users about the mistake location in their submitted query that contributed to the logic error. By this warning, hopefully users are able to apply correction and learn from their mistakes. This process makes a comparison between submitted SQL with expected SQL. Submitted SQL is the user answer's SQL to solve the related case and expected SQL is a key answer which is stored in data base. The differences between those data indicate the logic error position and will be used as warning to users. Because this research material is from SQL selection keyword, this warning is divided into three blocks warning, they are SELECT block, FROM block and WHERE block. SELECT block means the mistakes space select clause. This implies that the mistake is in the select column. FROM block warning will be shown if the mistake occurred nearby from clause. Wrong selected table is the example of this case. And WHERE block

will be shown if the mistake took placed in where clause. This occurs when there is a missed condition. This checking method compares submitted SQL and expected SQL sequentially. The idea is to make users do correction by the first mistake and then the rest. For example if there are logic error in select, from and where block, this system shows only the select position. It can be conclude that this system shows the first logic error.

III. Results and Discussion

This system is able to do prediction about user's mistake regarding SQL logic error. Therefore, the system evaluation is about to check the rate of precision, recall and accuracy. The evaluation of this prediction system is based on confusion matrix [5], [16]. This matrix compares between the prediction value and the real value. Confusion matrix is shown as Table 1.

There are four types of value that are produced by confusion matrix. True positive value shows that system gives correct prediction. In this system case, it gives correct logic error warning. False negative value shows that system gives incorrect prediction. In this research case, system gives free of logic error warning but it should be logic error warning. False positive is also incorrect prediction. System gives logic error warning but that should be free of logic error warning. Finally, true negative value shows correct result. System shows that there is no logic error and it is same with the real one.

From those values in confusion matrix, precision recall and accuracy rate can be calculated [17]. Precision rate compares between the quantities of correct system prediction with produced prediction by system. This value can be obtained by (1).

$$Precision = \frac{\text{TP}}{(\text{TP+FP})} x100\% \quad (1)$$

Recall value can be obtained by comparing between the quantities of correct system prediction with correct prediction should be. This value can be calculated by (2).

$$Recall = \frac{\text{TP}}{(\text{TP+FN})} x100\% \quad (2)$$

Accuracy value can be calculated by comparing the quantity between correct predictions by system with all system trial. It can be obtained by (3).

$$Accuracy = \frac{\text{TP+TN}}{TP+FN+FP+TN} x100\% \quad (3)$$

The system evaluation was conducted by giving input to the system and observing the output. Since this system has 36 cases about selection concept in SQL, the evaluation was done by 72 testing. For each case, there was a free logic error input and a logic error input. By those inputs, observation to the output was conducted then analysed whether it is correct or not. The output is the warning which is about the prediction of mistakes position in the user's SQL query. The syntax error detection is not evaluated because it is considered as a valid tool since it compiles SQL directly by DBMS (Data Base Management System) in web server. The examples of evaluation data is shown as Table 2.

Table 2 is the example of system evaluation which has 72 input queries. Each query was submitted to the system so system produced error warning which is prediction about occurred error mistake. Output prediction value column is the value of system prediction output. There are T (true) and F (false) values. The same kind of values are also in the logical value column which is the real value based on logical analysis. Those values was counted so that confusion matrix was produced as Table 3.

Table 1. Confusion matrix

		System Prediction Value	
		True	**False**
Real Value	True	True Positive (TP)	False Negative (FN)
	False	False Positive (FP)	True Negative (FN)

By the value in Table 3, precision, recall and accuracy values can be calculated using each formula as described before. Table 4 shows the percentages of precision, recall and accuracy. Those testing value in Table 4 show that system is able to detect logic error well. Precision rate shows 87.5% and it means that system is appropriate to detect logic error. Recall rate is 97.2% and it means that system is succeeded in logic error detection. And precision rate is 91.7% and it shows the system prediction and real prediction closeness.

Based on the obtained evaluation result, system is able to detect logic error well although it is not 100%. From 72 input queries, there were 6 mistake predictions. Those mistakes are shown as Table 5.

Table 3 shows the mistake prediction by system from 72 testing. These are divided into two types of mistake. There are 5 cases false negative and 1 false positive. As discussed earlier, false negative is occurred when system shows free of logic error but it should be logic error. Based on evaluation, it happened because the result table is same with expected table although the SQL is different. In this case, user SQL syntax has logic error. This can happen, because to detect logic error, system uses result table as the main source as described before. This type of case happened with the first five cases in Table 3. For example with the first case which is a logical error that occurred in WHERE clause. User's SQL is <= 101 and it should be <=100. Since both of those SQL syntaxes produce the same table result, system detects them as free of logic error. So, system does not check the differences between user's SQL and SQL key.

The second type of mistake is false positive. Based on the evaluation, system did false positive once and it is case number 6 in Table 3. System detects logic error but actually user's SQL is free of logic error. On that case, user answered "country" instead of "state". So system shows warning about logical error in SELECT clause. Logically, user should select "country", but key answer which is stored in database is "state". It makes the result tables of those SQL are different, then system gets it

Table 2. Examples of system evaluation

No	SQL query input	Output prediction value	Logical value
1	select * from department	T	T
2	select name from department	F	F
3	select city from department	T	T
4	select city from customer	F	F
5	select distinct country from customer	T	T
6	select country from customer	F	F
7	select * from employee where id=2;	T	T
8	select * from employee where id!=2;	F	F
9	select * from product where stock != 0	T	T
…	…	…	…
72	select name, now() from customer	F	F

Table 3. Evaluation result as confusion matrix

		System Prediction Value	
		True	**False**
Real Value	True	35	1
	False	5	31

Table 4. Evaluation results

Testing	Value
Precision	87.5%
Recall	97.2%
Accuracy	91.7%

Table 5. Mistake predictions

No	User's SQL	SQL Key	Warning
1	select * from product where stock <= 101	select * from product where stock <= 100	Free of logic error
2	select * from product where stock not between 10 and 15	select * from product where stock not between 10 and 20	Free of logic error
3	select id from department order by country desc	select emp_id from department order by country desc	Free of logic error
4	select country from customer union select distinct country from department	select country from customer union select country from department	Free of logic error
5	select avg(quantity)from detail_trans where id<5	select avg(quantity)from detail_trans	Free of logic error
6	select left(country,3) from customer	select left(state,3) from customer	There is logic error near SELECT clause

Table 6. Example of looping comparison

Trial	Total Character	Start End Mid	Brute Force
1	2701	2	89
2	621	2	195
3	840	2	527
4	2701	71	144
5	3279	2	1388
6	1035	188	984
7	3732	2	424
8	1665	2	221
9	3635	1360	1510
10	2486	2	814
...	...	...	...
50	1020	63	964

as logic error. This type of mistake happened because admin made mistake when storing the key answer to database.

As discussed before that Start End Mid algorithm is a developed algorithm based on Brute Force, this research also compare those algorithms. Both of them are simple algorithm which means they have no preprocessing phase. To know about the better algorithm in logical error detection, the looping of both algorithms was recorded. Those data is shown in Table 6.

Table 6 shows that there are 50 trials of logic error which detected by both of Start End Mid and Brute Force algorithm. In the first trial, there were 2701 total characters that will be checked in order to find the different data. The checking process was conducted by both of Start End Mid and Brute Force algorithm. Start End Mid found the different data on 2nd looping, but Brute Force found the different data on 89th looping. It means that Start End Mid is faster than Brute Force. But in the implementation, there was no significant difference in timing since this research compares not that much data. The compared character was lower than 10.000 characters. The rest trials had the same result that Start End Mid has lower looping in logical error detection process than Brute Force.

The result shows that Start End Mid was able to find the different data mostly in 2nd looping. It is because this algorithm checks the last character of data in the 2nd looping, so that the diffence can be found faster. It is different with Brute Force which cheks the first until the last character sequentially. If the different data is in the middle or even in the last, this algorithm will have more looping for checking process. In average data, the amount of total charactera were 1132 characters, but Start End Mid was able to find the different data on 61st looping. In the other hand, Brute Force was able to find the different data on 479th looping.

IV. Conclusion

Based on this research, obtained rate of precision, recall and accuracy show that system is able to detect logic error well. But there are still mistake predictions which are false negative and false positive. To avoid false negative prediction, the case that produces unique table should be made. It makes no other possible table which can produce same table by such SQL syntax. Besides that, to avoid false positive prediction, admin should give much attention to database design for storing free logic error SQL key.

The comparing result between Start End Mid and Brute Force algorithm shows that Start End Mid Algorithm has lower looping then Brute Force regarding to find the different character in such data. It means that Start End Mid Algorithm is faster than Brute Force in order to find the logical error which occured in user's SQL query.

References

[1] R. Dollinger, "SQL Lightweight Tutoring Module–Semantic Analysis of SQL Queries based on XML Representation and LINQ," in *EdMedia: World Conference on Educational Media and Technology*, 2010.

[2] S. Brass and C. Goldberg, "Semantic errors in SQL queries: A quite complete list," *J. Syst. Softw.*, vol. 79, no. 5, pp. 630–634, 2006.

[3] A. Ahadi, V. Behbood, A. Vihavainen, J. Prior, and R. Lister, "Students' Syntactic Mistakes in Writing Seven Different Types of SQL Queries and its Application to Predicting Students' Success," *Proc. 47th ACM Tech. Symp. Comput. Sci. Educ. - SIGCSE '16*, pp. 401–406, 2016.

[4] A. Fanani, "Pengembangan Sumber Belajar SQL Berbasis Web untuk Matakuliah Basis Data Prodi S1 Pendidikan Teknik Informatika Universitas Negeri Malang," Universitas Negeri Malang, 2014.

[5] E. P. Costa, A. C. Lorena, A. C. P. L. F. Carvalho, and A. A. Freitas, "A Review of Performance Evaluation Measures for Hierarchical Classifiers," in *AAAI-2007 Workshop, AAAI Technical Report WS-07-05*, 2007.

[6] C. Goldberg, "Do You Know Sql? About Semantic Errors in Database Queries," in *7th Workshop on Teaching, Learning and Assessment in Databases*, 2009.

[7] J. Ardiansah, O. Yasuhisa, and T. Wibawa, Aji Prasetya Widyaningtyas, "Development and Trial Use of a Web-based Database Learning System," in *JAiSE (Japanese Society for Information and System in Education)*, 2017.

[8] T. Lecroq, "Experimental results on string matching algorithmsitle," *Softw. Pract. Exp.*, vol. 25, no. 7, pp. 727–765, 1995.

[9] O. Masanori, T. Ryo, and S. Tadamasa, "An Evaluation of String Search Algorithms at User Standing," in *Proceedings of the 3rd WSES International Conference on Mathematics and Computers in Mechanical Engineering (MCME)*, 2001, pp. 4231–4236.

[10] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and S. Clifford, *Introduction to Algorithms 3rd edition*. MIT Press, 2009.

[11] R. A. Abdeen, "An Algorithm for String Searching Based on Brute-force Algorithm," *IJCSNS Int. J. Comput. Sci. Netw. Secur.*, vol. 11, no. 7, pp. 24–27, 2011.

[12] B. W. Watson and R. E. Watson, "A Boyer–Moore-style algorithm for regular expression pattern matching," *Sci. Comput. Program.*, vol. 48, no. 2–3, pp. 99–117.

[13] M. T. Goodrich and R. Tamassia, *Algorithm design*. Wiley, 2002.

[14] L. I. Zhulin, "A Method for Data Structure Course Design Based on CDIO Teaching Idea 2 The Course Design Based on CDIO Model," pp. 418–421.

[15] J. Ardiansah, O. Yasuhisa, and T. Wibawa, Aji Prasetya Widyaningtyas, "Developing of a Web-based Database Learning Support System for Practical Implementation of SQL," *IEICE (Institute Electron. Inf. Commun. Eng. Tech. Rep.*, vol. 116, no. 266, pp. 57–62, 2016.

[16] T. Saito and M. Rehmsmeier, "The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets," *PLoS One*, vol. 10, no. 3, 2015.

[17] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognit. Lett.*, vol. 27, no. 8, pp. 861–874, 2006.